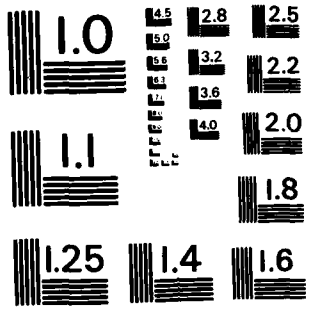
AD-A127 900 A DEDUCTIVE APPROACH TO THE DEBUGGING VERIFICATION AND MODIFICATION OF PROGRAMS(U) STANFORD UNIV CA DEPT OF COMPUTER SCIENCE Z MANNA OCT 82 AFOSR-TR-83-0335 1/1

UNCLASSIFIED AFOSR-81-0014 F/G 9/2 NL



A DEDUCTIVE APPROACH TO THE DEBUGGING, VERIFICATION, AND MODIFICATION OF PROGRAMS

by

Zohar Manna
Professor of Computer Science
Stanford University Stanford, CA 94305

Interim Scientific Report:
Air Force Office of Scientific Research
Grant AFOSR-81-0014
Oct. 1, 1981 - Sept. 30, 1982

Our research was concentrated on the following topics:

1. Verification of Concurrent programs: The Temporal Framework ([1]).

We first introduce temporal logic as a tool for reasoning about sequences of states. Models of concurrent programs based both on transition graphs and on linear-text representations are presented and the notions of concurrent and fair executions are defined.

The general temporal language is then specialized to reason about those execution sequences that are fair computations of a concurrent program. Subsequently, the language is used to describe properties of concurrent programs.

The set of interesting properties is classified into *invariance* (safety), *eventuality* (liveness), and *precedence* (until) properties. Among the properties studied are: partial correctness, global invariance, clean behavior, mutual exclusion, absence of deadlock, termination, total correctness, intermittent assertions, accessibility, responsiveness, safe liveness, absence of unsolicited response, fair responsiveness, and precedence.

2. Verification of Concurrent Programs: Temporal Proof Principles ([2]).

Here, we present temporal proof methods for establishing properties of concurrent programs. We consider three classes of properties: invariances, eventualities (liveness properties) and precedence (*until* properties).

The proof principle for establishing invariance properties is based on computational induction, and is a generalization of the *inductive assertions* method. For a restricted class of programs we present an algorithm for the automatic derivation of invariant assertions.

In order to establish eventuality properties we present several principles which translate the structure of the program into basic temporal statements about its behavior. These principles can be viewed as providing the temporal semantics of the program. The basic statements thus derived are then combined into temporal proofs for the establishment of eventuality properties. This method generalizes the method of *intermittent assertions*.

An *until* property is shown to be essentially a combination of a conditional invariance and an eventuality. Consequently the proof method for establishing an until property is a generalization of the method for establishing eventualities.

3. Verification of Sequential Programs: Temporal Axiomatization ([3]).

Earlier, we introduced temporal logic as a tool for reasoning about concurrent programs and specifying their properties ([1]) and presented proof principles for establishing these properties ([2]). Here, we restrict ourselves to deterministic, sequential programs. We present a proof system in which properties of such programs, expressed as temporal formulas, can be proved formally.

Our proof system consists of three parts: a *general* part elaborating the properties of temporal logic, a *domain* part giving an axiomatic description of the data domain, and a *program* part giving an axiomatic description of the program under consideration.

We illustrate the use of the proof system by giving two alternative formal proofs of the total correctness of a simple program.

4. Verification of Concurrent Programs: A Temporal Proof System ([4]).

A proof system based on temporal logic is presented for proving properties of concurrent programs based on the shared-variables computation model. As in [3], the system consists of three parts: the *general* uninterpreted part, the *domain* dependent part and the *program* dependent part. In the general part we give a complete system for first-order temporal logic with detailed proofs of useful theorems. This logic enables reasoning about general time sequences. The domain dependent part characterizes the special properties of the domain over which the program operates. The program dependent part introduces program axioms which restrict the time sequences considered to be execution sequences of a given program.

The utility of the full system is demonstrated by proving invariance, liveness and precedence properties of several concurrent programs. Derived proof principles for these classes of properties, are obtained which lead to compact representation of proofs.

The program dependent part is proved to be relatively complete. We then show that its dependence on the particular computation model studied is modular, by presenting a similar system for proving properties of CSP programs.

5. How to Cook a Temporal Proof System for General Languages ([5]).

An abstract temporal proof system is presented whose program-dependent part has a high-level interface with the programming language actually studied. Given a new language, it is sufficient to define the interface notions of atomic transitions, justice, and fairness in order to obtain a full temporal proof system for this language. This construction is particularly useful for the analysis of concurrent systems. We illustrate the construction on the shared-variable model and on CSP. The generic proof system is shown to be relatively complete with respect to pure first-order temporal logic.

6. Verification of Concurrent Programs: Proving Eventualities by Well-Founded Ranking ([6]).

We present proof methods for establishing eventuality and until properties. The methods are based on *well-founded ranking* and are applicable to both "just" and "fair" computations. These methods do not assume a decrease of the rank at each computation step. It is sufficient that

there exists one process which decreases the rank when activated. Fairness then ensures that the program will eventually attain its goal.

In the finite state case the proofs can be represented by diagrams. Several examples are given.

7. Synthesis of Communicating Processes from Temporal Specifications ([7],[8]).

We apply Propositional Temporal Logic (PTL) to the specification and synthesis of the synchronization part of communicating processes. To specify a process, we give a PTL formula that describes its sequence of communications. The synthesis is done by constructing a model of the given specifications using a tableau-like satisfiability algorithm for PTL. This model can then be interpreted as a program.

8. Deductive Synthesis of the Unification Algorithm ([9]).

The ***deductive approach*** is a formal program construction method in which the derivation of a program from a given specification is regarded as a theorem-proving task. To construct a program whose output satisfies the conditions of the specification, we prove a theorem stating the existence of such an output. The proof is restricted to be sufficiently constructive so that a program computing the desired output can be extracted directly from the proof. The program we obtain is applicative and may consist of several mutually recursive procedures. The proof constitutes a demonstration of the correctness of this program.

To exhibit the full power of the deductive approach, we apply it to a nontrivial example — the synthesis of a ***unification algorithm***. Unification is the process of finding a common instance of two expressions. Algorithms to perform unification have been central to many theorem-proving systems and to some programming-language processors.

The task of deriving a unification algorithm automatically is beyond the power of existing program synthesis systems. In this paper we use the deductive approach to derive an algorithm from a simple, high-level specification of the unification task. We will identify some of the capabilities required of a theorem-proving system to perform this derivation automatically.

9. Special Relations in Program Synthetic Deduction ([10]).

Program synthesis is the automated derivation of a computer program from a given specification. In the ***deductive approach***, the synthesis of a program is regarded as a theorem-proving problem; the desired program is constructed as a by-product of the proof. This paper presents a formal deduction system for program synthesis, with special features for handling equality, the equivalence connective, and ordering relations.

In proving theorems involving the equivalence connective, it is awkward to remove all the quantifiers before attempting the proof. The system therefore deals with ***partially skolemized sentences***, in which some of the quantifiers may be left in place. A rule is provided for removing individual quantifiers when required after the proof is under way.

The system is also ***nonclausal***; i.e., the theorem does not need to be put into conjunctive normal form. The equivalence, implication, and other connectives may be left intact.

Publications

[1] Z. Manna, A. Pnueli, "Verification of Concurrent Programs: The Temporal Framework", in *The Correctness Problem in Computer Science* (R.S. Boyer and J S. Moore, eds.), International Lecture Series in Computer Science, Academic Press, London (1981), pp. 215–273.

[2] Z. Manna, A. Pnueli, "Verification of Concurrent Programs: Temporal Proof Principles", Proc. of the Workshop on Logics of Programs (D. Kozen, ed.), Yorktown-Heights, N.Y. (1981). Springer-Verlag Lecture Notes in Computer Science 131, pp. 200–252.

[3] Z. Manna, "Verification of Sequential Programs: Temporal Axiomatization", in *Theoretical Foundations of Programming Methodology* (M. Broy and G. Schmidt, eds.), NATO Scientific Series, D. Reidel Pub. Co., Holland (1981), pp. 53–102.

[4] Z. Manna, A. Pnueli, "Verification of Concurrent Programs: a Temporal Proof System," Proc. 4th School on Advanced Programming, Amsterdam, Holland (June 1982).

[5] Z. Manna, A. Pnueli, "How to Cook a Temporal Proof System for Your Pet Language," in the Proc. of the Symposium on Principles of Programming Languages, Austin, Texas (Jan. 1983).

[6] Z. Manna, A. Pnueli, "Verification of Concurrent Programs: Proving Eventualities by Well-Founded Ranking," ACM Trans. on Programming Languages and Systems (to appear, 1983).

[7] Z. Manna, P. Wolper, "Synthesis of Communicating Processes from Temporal Specifications" *Proceedings of the Workshop on Logics of Programs (Yorktown-Heights)*, N.Y., Springer-Verlag Lecture Notes in Computer Science, 1981 (to appear also in the ACM Trans. on Programming Languages and Systems, 1983).

[8] P. Wolper, "Synthesis of Communicating Processes from Temporal Logic Specifications" Ph.D. Thesis, Computer Science Dept., Stanford University (August 1982).

[9] Z. Manna, R. Waldinger, "Deductive Synthesis of the Unification Algorithm", Science of Computer Programming, 1 (1981), pp. 5–48.

[10] Z. Manna, R. Waldinger, "Special Relations in Program Synthetic Deductions", Journal of the ACM (to appear, 1983).

A DEDUCTIVE APPROACH TO THE DEBUGGING, VERIFICATION, AND MODIFICATION OF PROGRAMS

by

Zohar Manna
Professor of Computer Science
Stanford University Stanford, CA 94305

Interim Report of Inventions:
Air Force Office of Scientific Research
Grant AFOSR-81-0014
Oct. 1, 1981 - Sept. 30, 1982

There are no patents or copyrights to report.

Zohar Manna

Zohar Manna

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER **AFOSR-TR- 83-0335** 2. GOVT ACCESSION NO. AD-A127 900	3. RECIPIENT'S CATALOG NUMBER
4. 'A DEDUCTIVE APPROACH TO THE DEBUGGING, VERIFICATION, AND MODIFICATION OF PROGRAMS'	5. TYPE OF REPORT & PERIOD COVERED Interim, 1 OCT 81-30 Sep 82
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Zohar Manna	8. CONTRACT OR GRANT NUMBER(s) AFOSR-81-0014
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Computer Science Stanford University Stanford CA 94305	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS PE61102F; 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS Mathematical & Information Sciences Directorate Air Force Office of Scientific Research Bolling AFB DC 20332	12. REPORT DATE OCT 82
	13. NUMBER OF PAGES 5
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.	
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)	
18. SUPPLEMENTARY NOTES	
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report summarizes research activities for the period 1 October 1981 to 30 September 1982 supported by the grant, and lists publication titles resulting from the research. Research was concentrated on the following topics: (1) Verification of Concurrent Programs: The Temporal Framework; (2) Verification of Concurrent Programs: Temporal Proof Principles; (3) Verification of Sequential Programs: Temporal Axiomatization; (4) Verification of Concurrent Programs: A Temporal Proof System; (5) How to Cook a Temporal Proof System for General Languages; (6) Verification of Concurrent Programs: →(CONTINUED)

DD FORM 1 JAN 73 1473

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ITEM #20, CONTINUED: Proving Eventualities by Well-Founded Ranking; (7) Synthesis of Communicating Processes from Temporal Specifications; (8) Deductive Synthesis of the Unification Algorithm; and (9) Special Relations in Program Synthetic Deduction.


ATE
LME